Multitask Learning for Object Localization With Deep Reinforcement Learning

Yan Wang, Lei Zhang, *Senior Member, IEEE*, Lituan Wang, *Student Member, IEEE*, and Zizhou Wang

***Abstract*—In object localization, methods based on a top-down search strategy that focus on learning a policy have been widely researched. The performance of these methods relies heavily on the policy in question. This paper proposes a deep *Q*-network (DQN) that employs a multitask learning method to localize class-specific objects. This DQN agent consists of two parts, an action executor part and a terminal part. The action executor determines the action that the agent should perform and the terminal decides whether the agent has detected the target object. By taking advantage of the capability of feature learning in a multitask method, our method combines these two parts by sharing hidden layers and trains the agent using multitask learning. A detection dataset from the PASCAL visual object classes challenge 2007 was used to evaluate the proposed method, and the results show that it can achieve higher average precision with fewer search steps than similar methods.**

***Index Terms*—Deep *Q*-network (DQN), multitask learning, object localization, reinforcement learning (RL).**

I. Introduction

Localization or detection of objects involve marking them out in images using landmarks, such as center points, contours, bounding boxes, and pixel-wise. Object localization is among the fundamental open problems in computer vision. It forms an inherent task in the domain of automatic understanding and interpretation of images, and subsequently, to analyze the spatial relations present among various objects in an image [1]. Object localization has thus been widely studied by researchers over the last few decades.

To localize object, the classic method is sliding windows approach [2]–[4]. This approach will traverse the whole image under different window sizes then using a classifier to classify each window. It can localize objects precisely but its time complexity is too high, generating a large number of redundant windows that increase a lot of work of the classifier. To reduce the number of windows, the Region-based Convolutional Network (R-CNN) detector was proposed by Girshick *et al.* [5]. It combines object proposal and Convolutional Neural Network (CNN) features to achieve state-of-the-art performance in Pascal and ImageNet benchmarks. Few thousands windows were preselected and classified. A series of proposals have subsequently been made based on the R-CNN that can all be seen as representative of a bottom-up search strategy method [6], [7], as they search for small patches of an image and merge them to get the final detection location. Another way to resolve the problem of complexity is the top-down search strategy methods [8]–[13]. The top-down search strategy methods usually learn the strategies by using recurrent neural networks (RNNs) or reinforcement learning (RL) methods. RNNs have been studied theoretically for many years [14]–[16], they are good at solving sequential problem. RL methods usually are used to learn policies. Methods falling under this strategy focus on such objects as biological vision systems, which are believed to follow a sequential process with changing retinal fixations that gradually accumulate evidence of certainty [17], [18].

Typically, the top-down search methods learn policies to search for an object by the sequential translation and reshaping of a bounding box. RL is an excellent means of learning policies by training an agent, that has achieve state-of-the-art performance in many fields [19], [20]. The deep *Q*-network (DQN) as a branch of RL has been used for object localization [10], [21]. Although these methods have reduced many detection regions compared to bottom-up strategy, it also has redundant steps. The performance of these methods relies heavily on the policies that the agents have learned. And the number of steps is significantly dependent on the feedback of accumulating rewards.

In this paper, following the top-down search strategy, in order to improve the performance of agent, we proposed an RL method with multitask learning to detect objects more precisely with fewer search steps. Specifically, the chosen RL method is the DQN, and the number of redundant steps can be considerably reduced by designing a new reward function. We consider the detector to be an agent, that uses eight actions to localize objects by rescaling the bounding box and a classifier for the terminal action, which determines when the agent should stop searching. Fig. 1 shows the framework of the entire localization process. The agent accepts an image and outputs an action and a terminal signal. If the terminal signal is "true," the entire search process is stopped and the region in the bounding box is what we localized. Otherwise,

Manuscript received February 9, 2018; revised October 30, 2018; accepted December 1, 2018. Date of publication December 10, 2018; date of current version December 9, 2019. This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFC0801800, in part by the National Natural Science Foundation of China under Grant 61332002 and Grant 61772353, in part by the Fok Ying Tung Education Foundation under Grant 151068, and in part by the Foundation for Youth Science and Technology Innovation Research Team of Sichuan Province under Grant 2016TD0018. *(Corresponding author: Lei Zhang.)*

The authors are with the College of Computer Science, Sichuan University, Chengdu 610065, China (e-mail: wangyan@stu.scu.edu.cn; leizhang@scu.edu.cn; wanglituan@stu.scu.edu.cn; 2017323040018@stu.scu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCDS.2018.2885813

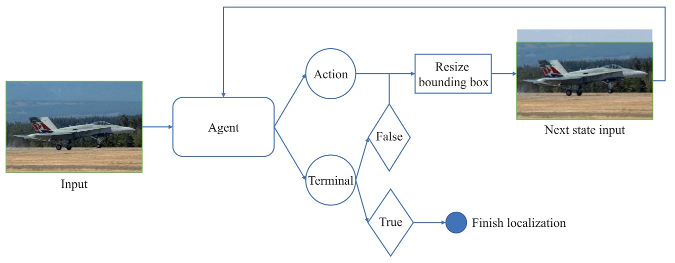


Fig. 1. Framework of the entire localization process. The detail neural network's architecture of agent is shown in Fig. 2. The action and terminal are the output of multitask learning network.

the bounding box is resized according to the action and the region in the bounding box is the input of agent in the next localization process.

The remainder of this paper is organized as follows. In Section II, we discuss related works. The details of deep Q-learning with multitask learning network architecture and algorithm are described in Section III. In Section IV, we detail experiments on the popular benchmark data sets PASCAL Visual Object Classes Challenge (VOC) 2007 and analyze the results to show how they verify our method. In the last, Section V, we conclude in this paper.

II. Related Works

Since it was proposed, RL has been widely used in a variety of fields. Google's DeepMind has been the most impressive proposal in the area in recent years-particularly its AlphaGo system [19], which was the first to defeat a human professional player at the board game Go. DeepMind was used to train an agent to play videogames on the Atari 2600 system, and achieved state-of-the-art results [20] that surpassed human performance. For image classification, neural networks have been studied theoretically for many years [14]–[16], [22], [23] and deep neural networks (DNNs) have been applied in many fields [5], [24], [25] due to the ability to automatically extract hierarchical features. The common ways are training neural networks to extract features especially CNNs [26]–[28], then following a classifier to classify these features. But recently, spatial glimpse policies have been learned for image classification [8], image caption [29], and activity recognition [30] by using RL, which has also been used for object detection. Mathe *et al.* [31] used the policy gradient method to learn policy to localize objects in images and this method mainly focused on speeding up the detection speed. However, this was at the expense of detection accuracy, and their method thus performed poorly on a public detection dataset.

Multitask learning is an approach to inductive transfer that improves generalization by using the domain information contained in the training signals of related tasks as an inductive bias. It does this by learning tasks in parallel while using a shared representation that can help other tasks to be learned better [32]. Multitask learning has been successfully used in a number of areas. Collobert and Weston [33] proposed a DNN method to enhance the performance of natural language processing (NLP) by dividing the whole NLP task into six standard subtasks and training by using multitask learning. In computer vision, Luo *et al.* [34] used multitask learning to learn more robust features. In order to improve generalization performance, multitask learning has been used to improve internal representation [35] by combining tasks from multiple-domain classification (for query classification) and information retrieval (ranking for Web search).

Recently, the research about object detection or localization are using CNN to extract features [6], [7], [36]–[39]. Besides, the most popular method in this vein is R-CNN [5] that, as a region proposal method, made a break through in Pascal VOC detection dataset. It proposes 2000 candidate regions then uses SVM to classify each region. A series of derived methods have been proposed based on the R-CNN. Fast R-CNN [6] uses single-stage training algorithm that jointly learned to classify object proposals and refine their spatial locations. Then Ren *et al.* [7] improved the R-CNN in terms of selecting region proposals by using a region proposal network to reduce proposal regions and proposed faster R-CNN. And there are other approaches such as YoLo [36] and MultiBox [40], etc. All these region proposal methods involve a large number of region proposals during detection. Different from region proposal methods, RL has been used to train agents to localize object, it can do so with fewer image regions to compute in the localization of each image. Based on RL, one way to localize objects is by refining the localization bounding box [31]. Caicedo and Lazebnik [10] used a class-specific active detection model by training a DQN agent to scale the bounding box and localize target objects. Instead of scaling the bounding box, Bellver *et al.* [11] trained an agent to select the correct region from five proposal boxes at each time step until the exact location of object is found. Jie *et al.* [12] proposed a tree-RL to solve detection task. When they evaluated their method on PACAL VOC dataset for object detection, they trained a tree-RL for region proposal, then trained a fast-rcnn for detection. It achieved similar performance as faster-rcnn on PASCAL VOC dataset. However, it has to train two networks separately instead of training an RL method end-to-end and the main role in the detection process is fast-rcnn. Yoo *et al.* [13] proposed an AttentionNet to detect objects and they trained the model combined PASCAL VOC dataset and ILSVRC 2012 dataset. In the experiment the authors chose two classes of PASCAL VOC dataset (human class and bottle class) instead of on the whole PASCAL dataset to evaluate. In all these methods, RL method reduced the number of proposal regions by modeling localization process into a sequential decision process. However, the performance of this kind of method depends on not only the policy learned by the agent, but also the terminal time. The agent usually stops early when the predict box is really close to the ground-truth bounding box but does not reach the positive boundary. So these methods cannot achieve high score on many detection bench mark detection datasets.

III. DQN Localization With Multitask Learning

The DQN is a kind of RL methods that uses a DNN to compute the Q-value, instead of learning the Q-table in Q-learning. Each output of DQN stand for the Q-value of each action. Each time the agent do the action which has the maximum Q-value of DNN.

A. Modeling Object Localization as Markov Decision Process

To use the DQN for object localization, the first step is to cast the problem of object localization as a Markov decision process (MDP). The MDP changes the localization process into a sequential decision process. The MDP works in the environment. In the localization task, considering an image as the environment, the agent transforms a bounding box using a set of actions and changes the environment state to the next one. The region in the final bounding box is that detected. We initialize the position of the bounding box on a board of the image. Usually, an MDP can be described by a four-tuple $(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{P}, \boldsymbol{R})$, where $\boldsymbol{S}$ represents a set of environment states, $\boldsymbol{A}$ is a set of actions, $\boldsymbol{P}$ represents the probability from one state translating to the other one, in this object detection task it determined by the Q-value, and $\boldsymbol{R}$ is a reward translation function that returns a reward while the given state is translated into the next one after the agent doing an action. It represents whether the agent doing this action is good. Each element in this MDP can be specifically set as follows.

1) *State Set* $\boldsymbol{S}$*:* Each image is represented as an environment. We use the feature vector of image $\boldsymbol{o}$ and concatenate it with the vector $\boldsymbol{h}$ of historical actions as a complete state $\boldsymbol{s} = <\boldsymbol{o}, \boldsymbol{h}>$. Image feature $\boldsymbol{o}$ is a 4096-D vector which is extracted by a pretrained CNN network. History $\boldsymbol{h}$ is a record of actions. We record eight recent steps to form a 1×8 vector. For each step, we use number zero-seven to represent the actions. Before concatenating feature vector $\boldsymbol{o}$ with action history vector $\boldsymbol{h}$, we encode each action into a one-hot code and flatten it to obtain a 1×64-dimensional history vector. The purpose of concatenating the history vector is to enable the agent to learn better from the past.

2) *Action Set* $\boldsymbol{A}$*:* In our action set, we define eight actions as [10] mentioned, namely *left, right, up, down, bigger, smaller, fatter,* and *taller*. These actions can be applied to the bounding box. Once the agent acts on each action, the bounding box is rescaled and the environment is transformed from the given state into a new one. A box is represented by the coordinates of the pixels of its two corners: $b = [x_1, y_1; x_2, y_2]$. The rescaling of bounding box is then achieved by making a discrete change to the box according to two factors which are defined as

$$\begin{cases} \beta_w = \beta \times (x_2 - x_1) \\ \beta_h = \beta \times (y_2 - y_1) \end{cases} \tag{1}$$

where β_w is a width factor, β_h is a height factor, and $\beta \in (0, 1)$ is a discount factor. For example, by performing the action *left*, the bounding box can be compute as $[x_1 - \beta_w, y_1; x_2 - \beta_w, y_2]$. In this paper, we set $\beta = 0.25$. In a complete MDP, the terminal action will be executed when the agent achieves its goal. We propose a network with softmax output to predict when the agent search should be terminated.

3) *Reward Function* $\boldsymbol{R}$*:* The return value of reward function R measures whether an action performs in the environment is good. In object localization, intersection-over-union (IoU) is used to measure the coverage between the target box and the predicted box. Let p be the predicted box and t the target box. The IoU between p and t can be written as $\mathrm{IoU}(p, t) = \mathrm{area}(p \cap t)/\mathrm{area}(p \cup t)$. The reward function can thus be constructed according to the IoU. Define the reward function $R(a, s, s')$ to represent the magnitude of the reward earned by the agent getting from state s after action a to transform to the next state s'. According to the computation of IoU, if action a is one of the eight actions, R can be written as

$$R\big(a, s, s'\big) = \begin{cases} 0, & \mathrm{sign}\big(\mathrm{IoU}\big(p', t\big) - \mathrm{IoU}(p, t)\big) > 0 \\ -1, & \text{otherwise} \end{cases} \tag{2}$$

where p' is the predicted bounding box at the next state s'. If action a is the terminal action, it is written as T, and we set the terminal reward as

$$R\big(T, s, s'\big) = \begin{cases} 8, & \mathrm{IoU}(p, t) \geq 0.5 \\ -8, & \text{otherwise.} \end{cases} \tag{3}$$

B. DQN With Multitask Learning for Object Localization

In order to solve the problem of inaccurate terminal times, we propose the DQN with multitask learning for localization. Multitask learning is the procedure of learning several tasks at the same time with the aim of mutual benefit. It can improve learning efficiency and prediction accuracy for task-specific models. We divide the terminal action and the eight bounding box operations into two networks with sharing hidden layers. These two networks addressed different but relevant tasks.

1) DQN With Multitask Learning Architecture: The fact of using Q-learning to solve localization problem is to use a Q-network to learn a policy to find the desired object in an image. In Q-learning, policy is the agent implements a mapping from states to probabilities of selecting possible actions. It is usually denoted by π. Q-learning learns policy by using Q-function (also called the action-value function), $Q_\pi(s, a)$, as the expected return starting from s by taking action a, and following the policy π. It can be computed as

$$Q_\pi(s, a) = \mathbb{E}[R_t | S_t = s, A_t = a] \tag{4}$$

where the subscribe t denotes time step. In order to learn the Q-function, the optimal action-value function is used, which obeys the Bellman optimality equation

$$Q_*(s, a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q_*\big(s', a'\big)\right] \tag{5}$$

where $r = R(a, s \to s')$ is the immediate reward for an agent performing action a at state s that is translated into state s'. $\gamma \in [0, 1]$ is a discount factor for future rewards.

Deep Q-learning [20] uses a DNN to represent the Q-function, which can be altered from $Q(s, a)$ to $Q(s, a; \theta)$, where θ represents the parameters of the DQN. By learning a set of suitable parameters θ, $Q(s, a; \theta)$ converges to the optimal value. The Q-network typically uses a fully connected

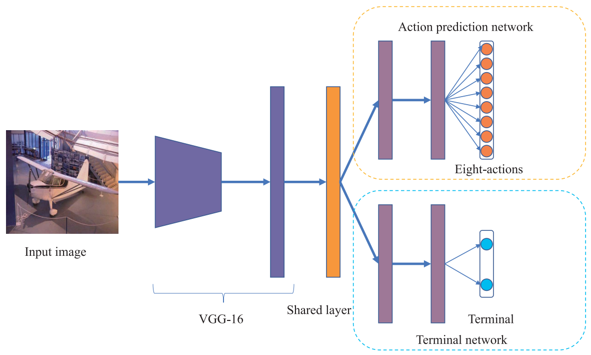


Fig. 2. Architecture of our network. The input images are resized as 224×224 pixels followed by a VGG-16 neural network for feature extraction. The output of this neural network is concatenated with the action history vector to the multitask Q-network.

neural network. In our model, we used a three-layer fully connected neural network where the first layer was shared with the softmax network. The architecture of our model is illustrated in Fig. 2.

In our model, the neural network is a pretrained VGG-16 network. It is trained for IMAGENET dataset classification, so we used the last fully connected layer as the feature layer entered into the Q-network. The first layer is the shared layer that shares the parameters between the action prediction network and the terminal network. The terminal network is a two-layer fully connected network where each hidden layer has 1024 units and the output layer is a binary classification softmax layer. The composition of the action prediction network is identical to that of the terminal network except that the output is eight units represent the Q-value of each action.

2) Training the Entire Network: There are two kinds of loss functions in our network: 1) a cross-entropy loss function for softmax output and 2) a mean square error loss function for Q-value output. The cross-entropy and the mean square error loss functions are defined, respectively, as

$$L_s(\theta) = -\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log\left(\hat{y}_n\right) + (1-y_n)\log\left(1-\hat{y}_n\right)\right] \quad (6)$$

$$L_c(\theta) = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{y}_n - y_n\right)^2 \quad (7)$$

where θ is the parameter of the network, N is the number of samples, y_n is the nth target label, and $\hat{y}_n$ is the nth predicted value. In multitask learning, there are many ways to combine these two loss functions. The most common way is to sum every part of each, we used a balance factor γ to balance two parts losses. Although it is a simple way, it works well. Thus, the joint loss function used here is

$$\begin{aligned} L(\theta_s,\theta_c) &= \gamma L_s(\theta_s) + (1-\gamma)L_c(\theta_c) \\ &= -\gamma\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log\left(\hat{y}_n + (1-y_n)\log\left(1-\hat{y}_n\right)\right]\right. \\ &\quad + (1-\gamma)\frac{1}{N}\sum_{n=1}^{N}\left(\hat{y}_n - y_n\right)^2 \end{aligned} \quad (8)$$

where θ_s denotes parameters in terminal network and θ_c denotes parameters in action prediction network. We set balance factor $\gamma = 0.6$ to make terminal network achieve better performance.

We used the back propagation algorithm to compute the gradient and update the parameters of the entire network using stochastic gradient descent (SGD). In the Q-learning framework, we used replay memory to stabilize the entire training process. The replay memory restores the current action a_t, state s_t, reward r_t, the next state s_{t+1} after act action s_t and the region label l_t of current state. It is represented as a five-tuple $<s_t, a_t, r_t, s_{t+1}, l_t>$. During the training process, ϵ-greedy algorithm is used to balance the agent's exploration and exploitation according to ϵ. The agent explores with a probability of ϵ and performs exploitation with a probability of $1-\epsilon$. In typical exploration, the agent randomly selects actions to observe different transitions and collects a varied set of experiences. To enable the agent to learn faster, we use the guided exploration strategy mentioned in [10]. In the course of exploitation, the agent selects actions according to the output of the Q-network and selects actions that maximum the Q-value. During the training process, ϵ will decreases from 1.0 to 0.1 with each set of 100 episodes reducing the value by 0.01. The algorithm is shown as Algorithm 1.

IV. Experiment

To evaluate the proposed multitask learning with the deep Q-learning method, the Pascal VOC2007 object detection dataset was used. VOC2007 dataset was widely used in detection. In top-down search strategy detection region, it is main dataset that used to evaluate all these methods [10], [11], [21]. The greater the amount of training data, the better performance of the neural network. Since we combined trainset of VOC2007 and VOC2012 dataset as our train dataset. The VOC dataset contained 20 classes. We first chose chose three classes—airplane, person, and bicycle to illustrate that our method is independent of pretrained feature extraction model. Then compared our method with baseline methods [5], [10], [40] in the whole VOC2007 test set.

A. Details of Implementation

The architecture of our method is shown in Fig. 2. We used a pretrained VGG-16 and AlexNet neural networks for extracting features. The history vector $\boldsymbol{h}$ concatenated with this fully connected layer was the next layer's input. During the experiment we found that the agent terminated search after approximately eight steps in each localization process. We thus recorded eight actions from recent history, which meant that vector $\boldsymbol{h} \in \mathbb{R}^{8\times 8}$. When trained the whole network, we used the SGD with learning rate $\alpha = 0.0001$ in tensorflow.

During the training process, we found that the agent usually stopped after fewer than 100 steps. Thus, instead of setting the maximum number of steps taken by the agent to 200, as in [10], we set to 100. During exploitation in the experiment, we found that the classifier could not classify the samples precisely when their IoU values were around 0.5. In particular, negative samples with $0.3 < \text{IoU} < 0.5$, were easily to

Algorithm 1: Multitask With Deep Q-Learning Algorithm

```
1  Initialize replay memory $\mathcal{D}$ to capacity of $n$;
2  Initialize all $Q$-network parameters $\theta$ and target parameters $\theta'$ as the same;
3  Initialize update step $u = 256$, replay memory index $i = 0$, $\epsilon = 1.0$, Maximum episode $M = 20000$;
4  for $episode = 1{:}M$ do
5      Reset environment, history;
6      if $i > n$ then
7          $i = i \mod n$;
8      else
9          continue;
10     end
11     if $\epsilon < 0.1$ then
12         $epsilon = 0.1$
13     end
14     Initialize state $s_1^i$;
15     for $t = 1{:}T$ do
16         Select a random action $a_t^i$ with probability $\epsilon$;
17         Otherwise select $a_t^i = \arg\max_a Q_a(a, s_t^i; \theta)$;
18         Execute action $a_t^i$ in emulator and observe immediately reward $r_t^i$ and softmax label $l_t^i$;
19         if $l_t^i$ is True then
20             Set $s_{t+1}^i = s_t^i$;
21             Set $r_t^i$ equal to terminal reward as shown in Eq. 3
22             Store a 5-tuple $< s_t^i, a_t^i, r_t^i, s_{t+1}^i, l_t^i >$ into replay memory
23         else
24             Set $s_{t+1}^i$ with $s_t^i, a_t^i$;
25             Store a 5-tuple $< s_t^i, a_t^i, r_t^i, s_{t+1}^i, l_t^i >$ into replay memory $\mathcal{D}$;
26         end
27         Sample a mini-batch for $< s_{t'}, a_{t'}, r_{t'}, s_{t'+1}, l_{t'} >$ from replay memory $\mathcal{D}$ randomly;
28         Use gradient decent to minimize loss function Eq. (8) and update parameter $\theta$;
29         if $(episode \mod u) == 0$ then
30             Copy training parameters $\theta$ to target parameters $\theta'$;
31         else
32             Continue;
33         end
34     end
35     if $(episode \mod 100) == 0$ then
36         $\epsilon = \epsilon - 0.1$
37     end
38 end
```

classify as positive. To solve this problem, we used the hard-negative mining method proposed by Felzenszwalb *et al.* [41] to train the classifier. In hard-negative method, hard negative samples were hence collected as part of a new negative training set and combined with the positive training set to train the classifier once again. In the experiment, we regarded the samples such that $0.4 \leq \text{IoU}(t, g) < 0.5$ as hard negative samples.

TABLE I
Two Pretrained CNNs Evaluate on Partly VOC2007 Data Set

dataset	method	aeroplane	person	bike
VOC2007	MultiBox [40]	41.3	37.4	27.7
	Caicedo et al. [10]	55.5	45.7	61.9
	Kong et al. [21]	-	45.6	**63.9**
	Ours(Alexnet)	58.4	46.2	62.5
	Ours(VGG-16)	**59.2**	**46.5**	62.3

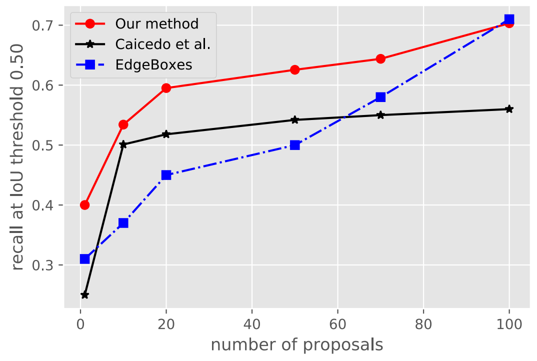


Fig. 3. Recall as function of the number of proposed regions. Solid lines are DQN localization work, red line is this paper, black line is the method proposed by Caicedo and Lazebnik [10]. Dashed line is a baseline work, named as edgeboxes.

After training for 15 000 episodes, we stopped adding all the negative samples to the replay memory and only added hard negative samples.

B. Performance Analysis

1) Evaluation of Precision: We compared performance of the proposed method with other methods recently proposed in the literature, including one top-down strategy works, the method proposed in [10]. Our method is inspired by Caicedo and Lazebnik [10], so this method is our main baseline. We also used two other baseline methods, MultiBox [40] and R-CNN [5]. The method proposed in [10] was the first method to use DQN to handle object localization. MultiBox [40] used deep CNN to predict bounding boxes containing objects and the score of each bounding box. This method predicts only a few bounding boxes. Our approach aims to localize objects in a small number of regions as well but, by combining DQN and multitask learning, learns a policy to select regions much more precisely. Caicedo and Lazebnik [10] used pretrained AlexNet for feature extraction, so we first evaluated our method in two different pretrained CNNs, namely AlexNet and VGG-16. For better comparison, we added another method the collaborative multiagent deep RL algorithm [21] which also used AlexNet for feature extraction. The collaborative multiagent deep RL algorithm [21] need localize two category at the same time. The authors chose those images that contained both bicycle and person as their training dataset. So we only compared this method on the selected three classes to evaluate the affection of feature extraction module. The results list in Table I. From Table I,

TABLE II
AP PER CATEGORY IN THE PASCAL VOC 2007 TEST SET

method	areoplane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	MAP
MultiBox [40]	41.3	27.7	30.5	17.6	3.2	45.4	36.2	53.5	6.9	25.6	27.3	46.4	31.2	29.7	37.5	7.4	29.8	21.1	43.6	22.5	29.2
Caicedo et al. [10]	55.5	61.9	38.4	36.5	21.4	56.5	58.8	55.9	21.4	40.4	46.3	**54.2**	56.9	**55.9**	45.7	21.1	47.1	41.5	54.7	**51.4**	46.1
Ours(AlexNet)	58.4	**62.5**	**41.2**	40.6	14.3	56.5	62.5	**58.5**	22.6	52.6	48.2	53.7	**58.6**	53.6	46.2	29.2	47.1	42.7	**55.2**s	48.5	47.6
Ours(VGG-16)	**59.2**	62.3	40.2	**41.2**	**24.1**	**59.5**	**67.1**	55.6	**24.1**	**64.1**	**50.2**	54.1	57.1	54.7	**46.5**	**29.4**	**48.5**	**44.2**	54.1	35.9	**48.6**
R-CNN [5]	64.2	69.7	50.0	41.9	32.0	62.6	71.0	60.7	32.7	58.5	46.5	56.1	60.6	66.8	54.2	31.5	52.8	48.9	57.9	64.7	54.2

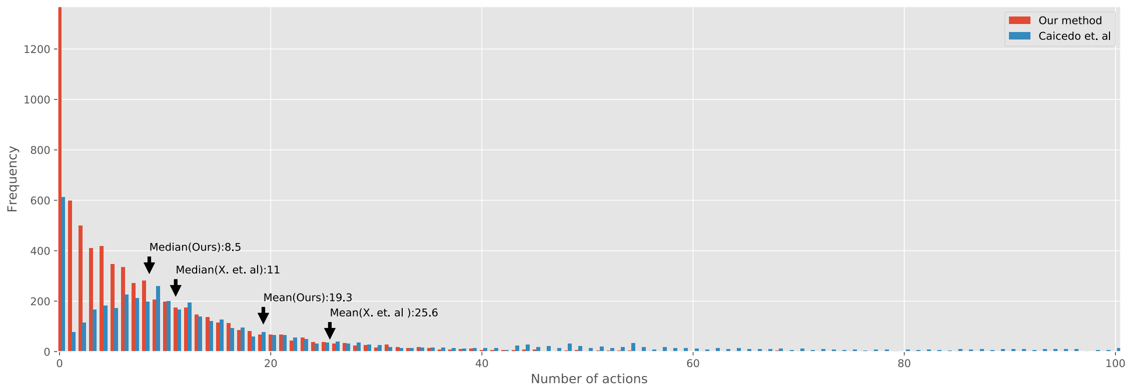


Fig. 4. Distribution of localization explicitly marked by the agent as a function of the number of actions required to reach the object on the whole VOC2007 dataset. The red bar is the steps distribution of our method, blue bar is the steps distribution of the method proposed by Caicedo and Lazebnik [10]

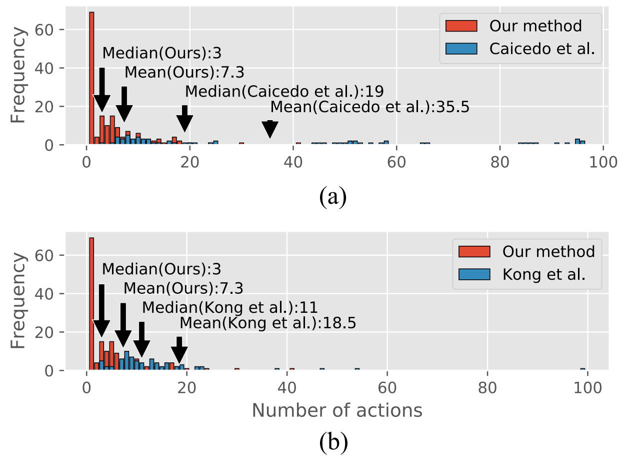


Fig. 5. Distribution of localization explicitly marked by the agent as a function of the number of actions required to reach the object. Distribution of (a) our method and the method proposed by Caicedo and Lazebnik [10] and (b) our method and collaborative multiagent.

we can see our method can achieve better performance than the baseline top-down strategy method and got almost the same performance as [21] whether used pretrained AlexNet or VGG-16. And the results also showed that two different pretrained networks had little effect on the localization results, but the pretrained VGG-16 network got a slightly better result.

Further, we evaluated our method which based on both pretrained AlexNet and VGG-16 on the whole PASCAL VOC2007 test set. The results tested on VOC2007 test set list in Table II. Our results were correspondingly significantly better than other similar baseline works as shown in Table II. Table II lists a detailed per-category average precision (AP) of the methods mentioned above. Compared with two similar works, by using multitask learning, our approach achieve a much better result. On the whole, the R-CNN yield the best performance and remained as a strong baseline. Its performance benefited from a large number of object proposals in the R-CNN that, however, demand significant computing power. R-CNN usually proposes one thousand to two thousand regions as the input of CNN, whereas we only generated approximately ten regions in each instance of detection. Thus, our method has reduced ten times the data that needed to be computed.

2) *Evaluation of Recall:* In the active search localization methods, all the regions attended by the agent can be looked as object proposal candidates. We followed the evaluation methods in [42] and compared with the active detection model proposed by Caicedo and Lazebnik [10] and the edge-box proposed by Zitnick and Dollár [43]. From Fig. 3, we see that our method can achieved higher recall values than the method proposed in [10] in which claimed they achieved state-of-the-art recall and edge-box.

To illustrate this result further, we presented the distribution of steps that localized object successfully on the whole VOC2007 dataset in Fig. 4. In Fig. 4, red bar is the steps distribution of our method and blue bar is the steps distribution of Caicedo's method. As we can see, the distribution of steps of our method got smaller both mean number of steps and median number of steps. As is evident, our approach significantly reduced the number of steps need for the agent to find the object. In particular, our method needed fewer than nine steps for most situations but Caicedo's method needs 11 steps. And the mean number of steps to localization object reduced from 25.6 to 19.3 when compared to Caicedo's method. There

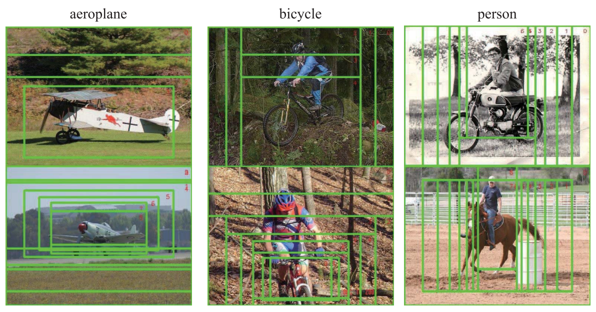


Fig. 6. Examples of successful localization. The agent stopped to search for the object when the current IoUs were around 0.5.

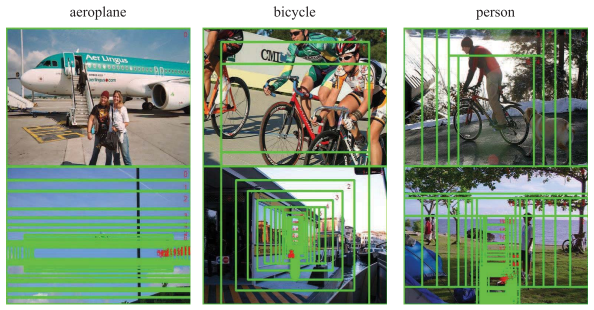


Fig. 7. Examples of failure localization. The first row shows the failure cases where the agent stopped too early. The second row shows the examples of where the agent repeated mutually exclusive actions.

are two reasons for our model to achieve higher recall with fewer steps. First, we used a terminal network to determine terminal time, and this was more powerful than an terminal action mentioned in [10]. Furthermore, the terminal network and action predict network were trained with multitask learning that enable the agent to learn a more robust policy. Second, the reward function shown in

$$R(a, s, s') = \begin{cases} 1, & \text{if } \operatorname{sign}\big(\mathrm{IoU}(t', g) - \mathrm{IoU}(t, g)\big) \geq 0 \\ -1, & \text{else.} \end{cases} \tag{9}$$

and used in [10] shows that if the agent performs a positive action, it receives a reward of $+1$ even if this action is a worse positive action. This situation causes the agent to learn to choose the worse positive action for achieving the maximal accumulate reward. By using our reward function in (2), however, our agent only takes several tens of steps to locate objects. It thus considerably reduce the number of required steps compared with [10]. Furthermore, we displayed the distribution of steps that localized object successfully on one specific class "bicycle" in Fig. 5. From Fig. 5, we can see that our method also got a smaller mean and median number of steps to localize objects than the two baseline top-down strategy works proposed in [10] and [21] on the bicycle class.

3) Qualitative Evaluation: We present a number of examples that the agent localized object on the Pascal VOC detection data set, and found that our method can localize most number of images. Fig. 6 shows some successfully localized examples and Fig. 7 presents instance of failure. From Fig. 6, one can see that the most successfully localized objects' bounding boxes are not strictly around the objects, and their IoU values are around 0.6. Because we combined multitask and hard-negative methods to train the terminal classifier, the classifier was sensitive to terminating the localization process when the values of the IoU were just localized more than 0.5. There were two kinds of failure situations. One is shown in the first row, where the agent stopped too early. These cases formed a small part of all instances of failure, caused by incorrect classification. The second kind of failure situation is repeating two mutually exclusive action and cannot jump out it until reach the maximum of step, as shown in the second row. This was caused by the use of the region in the bounding box as the input of the pretrained CNN. When the agent missed the region occupied by the object, the agent could not obtain the features of the object. Thus, combining the global features of the image with local features may be a good way to improve the localization process.

V. Conclusion

In this paper, we proposed a method to localize object that combine DQN and multitask learning. Following the training with multitask learning, the agent learned a more powerful policy that could locate the object more precisely. Furthermore, by designing a new reward function, our method reduced the number of search steps required of the agent in each instance of the localization. The evaluation results on the Pascal VOC2007 dataset showed that our method can reach considerably higher AP and localize most objects within fewer search steps, i.e., the number of median steps was 3 and the number of mean steps was 7.3. Experimental results verified the effectiveness of the proposed method.

References

[1] S. Choudhuri, N. Das, R. Sarkhel, and M. Nasipuri, "Object localization on natural scenes: A survey," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 32, no. 2, 2017, Art. no. 1855001.
[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, 2005, pp. 886–893.
[3] P. Felzenszwalb, D. Mcallester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2008, pp. 1–8.
[4] T. Malisiewicz, A. Gupta, and A. A. Efros, "Ensemble of exemplar-SVMs for object detection and beyond," in *Proc. Int. Conf. Comput. Vis. (ICCV)*, 2011, pp. 89–96.
[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 580–587.
[6] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1440–1448.
[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.
[8] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, "Recurrent models of visual attention," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2204–2212.
[9] A. Gonzalez-Garcia, A. Vezhnevets, and V. Ferrari, "An active search strategy for efficient object class detection," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3022–3031.
[10] J. C. Caicedo and S. Lazebnik, "Active object localization with deep reinforcement learning," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2488–2496.
[11] M. Bellver, X. Giro-i Nieto, F. Marques, and J. Torres, "Hierarchical object detection with deep reinforcement learning," in *Proc. Deep Reinforcement Learn. Workshop NIPS*, Dec. 2016, pp. 1–9.

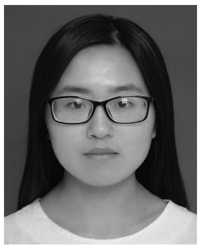
[12] Z. Jie *et al.*, "Tree-structured reinforcement learning for sequential object localization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 127–135.
[13] D. Yoo, S. Park, J.-Y. Lee, A. S. Paek, and I. S. Kweon, "AttentionNet: Aggregating weak directions for accurate object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2659–2667.
[14] L. Zhang and Z. Yi, "Dynamical properties of background neural networks with uniform firing rate and background input," *Chaos Solitons Fract.*, vol. 33, no. 3, pp. 979–985, Aug. 2007.
[15] L. Zhang, Z. Yi, and J. Yu, "Multiperiodicity and attractivity of delayed recurrent neural networks with unsaturating piecewise linear transfer functions," *IEEE Trans. Neural Netw.*, vol. 19, no. 1, pp. 158–167, Jan. 2008.
[16] L. Zhang and Z. Yi, "Selectable and unselectable sets of neurons in recurrent neural networks with saturated piecewise linear transfer function," *IEEE Trans. Neural Netw.*, vol. 22, no. 7, pp. 1021–1031, Jul. 2011.
[17] L. Itti, G. Rees, and J. K. Tsotsos, *Neurobiology of Attention*. Amsterdam, The Netherlands: Academic, 2005.
[18] H. Larochelle and G. E. Hinton, "Learning to combine foveal glimpses with a third-order Boltzmann machine," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 1243–1251.
[19] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
[20] V. Mnih *et al.* (2013). *Playing Atari With Deep Reinforcement Learning*. [Online]. Available: https://arxiv.org/abs/1312.5602
[21] X. Kong, B. Xin, Y. Wang, and G. Hua, "Collaborative deep reinforcement learning for joint object search," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1695–1704.
[22] L. Zhang, Z. Yi, S. L. Zhang, and P. A. Heng, "Activity invariant sets and exponentially stable attractors of linear threshold discrete-time recurrent neural networks," *IEEE Trans. Autom. Control*, vol. 54, no. 6, pp. 1341–1347, Jun. 2009.
[23] L. Zhang, Z. Yi, and S.-I. Amari, "Theoretical study of oscillator neurons in recurrent neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5242–5248, Nov. 2018.
[24] L. Wang, L. Zhang, and Z. Yi, "Trajectory predictor by using recurrent neural networks in visual tracking," *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3172–3183, Oct. 2017.
[25] Y. Chen, L. Zhang, and Z. Yi, "Subspace clustering using a low-rank constrained autoencoder," *Inf. Sci.*, vol. 424, pp. 27–38, Jan. 2018.
[26] K. Simonyan and A. Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. [Online]. Available: https://arxiv.org/abs/1409.1556
[27] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.
[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[29] K. Xu *et al.*, "Show, attend and tell: Neural image caption generation with visual attention," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2048–2057.
[30] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei, "End-to-end learning of action detection from frame glimpses in videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2678–2687.
[31] S. Mathe, A. Pirinen, and C. Sminchisescu, "Reinforcement learning for visual object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2894–2902.
[32] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, Jul 1997. [Online]. Available: https://doi.org/10.1023/A:1007379606734
[33] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 160–167.
[34] Y. Luo, Y. Wen, D. Tao, J. Gui, and C. Xu, "Large margin multi-modal multi-task feature extraction for image classification," *IEEE Trans. Image Process.*, vol. 25, no. 1, pp. 414–427, Jan. 2016.
[35] X. Liu *et al.*, "Representation learning using multi-task deep neural networks for semantic classification and information retrieval," in *Proc. HLT-NAACL*, 2015, pp. 912–921.
[36] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.
[37] G. Cheng, P. Zhou, and J. Han, "Learning rotation-invariant convolutional neural networks for object detection in VHR optical remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 12, pp. 7405–7415, Dec. 2016.
[38] J. Han, D. Zhang, G. Cheng, L. Guo, and J. Ren, "Object detection in optical remote sensing images based on weakly supervised learning and high-level feature learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 6, pp. 3325–3337, Jun. 2015.
[39] J. Han, D. Zhang, G. Cheng, N. Liu, and D. Xu, "Advanced deep-learning techniques for salient and category-specific object detection: A survey," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 84–100, Jan. 2018.
[40] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 2155–2162.
[41] P. F. Felzenszwalb, R. B. Girshick, D. Mcallester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.
[42] J. Hosang, R. Benenson, P. Dollár, and B. Schiele, "What makes for effective detection proposals?" *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 4, pp. 814–830, Apr. 2016.
[43] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 391–405.

Yan Wang is currently working toward the Ph.D. degree with the Machine Intelligence Laboratory, College of Computer Science, Sichuan University, Chengdu, China.

His current research interests include deep neural networks and object detection.

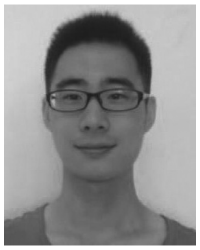
Lei Zhang (M'09–SM'17) received the B.S. and M.S. degrees in mathematics and the Ph.D. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2002, 2005, and 2008, respectively.

She was a Postdoctoral Research Fellow at the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong, from 2008 to 2009. She is currently a Professor at Sichuan University, Chengdu. Her current research interests include theory and applications of neural networks based on neocortex computing and big data analysis methods by very deep neural networks.



Lituan Wang (S'15) is currently working toward the Ph.D. degree with the Machine Intelligence Laboratory, College of Computer Science, Sichuan University, Chengdu, China.

Her current research interests include neural networks and visual tracking.



Zizhou Wang is currently working toward the Ph.D. degree with the Machine Intelligence Laboratory, College of Computer Science, Sichuan University, Chengdu, China.

His current research interests include neural network and object detection.